# Project 4: Two-phase Commit for Group Photo Collage

This project has two JAVA classes - Server.java and UserNode.java.
The Server initiates a two phase commit procedure, letting the users that contributed images to the collage examine it to see if they are happy with the result. UserNodes either approve or disapprove it. Only if all UserNodes that contribute an image to the collage approve, the collage is published (written to the Server's working directory).

The server maintains a global **transactionTable** HashMap that maps transactionID to transactionObj. This table maintains all details about each unique candidate collage image submitted for commit.
transactionObj contains -

- transactionID
- total number of Prepare messages sent
- name of collage
- collage image bytes
- isCommitted - if img is committed
- sourceTable - HashMap maps each contributing userNode to it's contributing images

The server updates logs on the dist in a file named **logFile**. Everytime the server starts and if the log file is present on disk, it updates its transactionTable. It sends DECISION message for all transactions in the log. The sendDecision() method ensures that the server sends the DECISION message only to those userNodes that have not ACK-ed the decision message previously.

The server sends two types of messages - PREPARE message and DECISION message to userNode.
PREPARE message has the following format -

```
<type><msg_body_len><msg_body><collage_img>
```

where, <msg_body> format is -

```
 <transactionID>:<contributing_img_name,contributing_img_name,...>
<type> = 1
```

DECISION message has the following format -

```
<type><msg_body>
```

where, <msg_body> format is -

```
<commit/abort>:<contrb_img_list,...>:<transactionID>
<type> = 2
```

When a new commit request is raised, startCommit() method is called. This method creates a new transactionObj record, updates the transactionTable and appends a log to the logFile on disk. It also generates the PREPARE message and sends it to each contributing userNode. It

executes a thread for each prepare message sent that waits for 6.5 secs and checks if the client responded to the prepare message. If the client has not, this is considered as a vote - NO.
If the all client vote YES, the collage is written to the Server's working directory. And the logFile is updated with the committed status of the transaction.
Once a abort or commit decision is made, the DECISION message is sent to all the nodes.
A new thread is spawned to send each DECISION messages. The thread sleeps for 6.5 seconds before it checks for an ACK by the userNode. The thread loops until it receives ACK from userNode.

UserNode.java
This class maintains a globals HashMap named imgBlockedTable. This contains mapping of imageName to it's status, that is, 2 - voted YES and waiting for a decision and 3 - deleted.
If a PREPARE message is received for any of the images present in the table, the userNode's vote is NO.
The userNode sends 2 types of message - **VOTE** and **ACK**.

VOTE message have the following format -
<message_type>:<transactionID>:<vote>
message_type : 1 - Vote ; 2 - Ack
Vote : 1 - Yes; 0 - NO
ACK message have the following format -
<message_type>:<YES>:<transactionID>
message_type : 1 - Vote ; 2 - Ack

If the DESICION message status is commit, it deletes the image from its working directory and updates the imgBlockTable accordingly. If the status is abort it simply removes the image entry form the table. These changes are logged in the log file on the disk immediately.
Every time a userNode is started, it checks if the logFile is present in the disk. If it is present it builds the imgBlockTable using the logs.
This way the userNode can receive the DECISION message any number of times, it will perform its appropriate task only once.

Key Designe Features:

- Every time a server is restarted all non-committed transaction are aborted
- The DECISION message is sent to userNodes that did not send the ACKS previously
- Each DECISION message is sent through a dedicated thread that loops until it receives an ACK
- Each PREPARE message has a dedicated thread that checks for user's vote after 6.5 seconds. Delayed or lost vote mean NO and lead to abort